Learning to play 3x3 games: neural networks as bounded-rational players

Sgroi, Daniel; Zizzo, Daniel John

Postprint / Postprint
Zeitschriftenartikel / journal article

Title: Learning to Play 3x3 Games: Neural Networks as Bounded-Rational Players

Authors: Daniel Sgroi, Daniel John Zizzo

PII: S0167-2681(08)00195-9
DOI: doi:10.1016/j.jebo.2008.09.008
Reference: JEBO 2261

To appear in: *Journal of Economic Behavior & Organization*

Received date: 12-4-2006
Revised date: 22-9-2008
Accepted date: 26-9-2008

Please cite this article as: Sgroi, D., Zizzo, D.J., Learning to Play 3x3 Games: Neural Networks as Bounded-Rational Players, *Journal of Economic Behavior and Organization* (2008), doi:10.1016/j.jebo.2008.09.008

Learning to Play 3×3 Games: Neural Networks as Bounded-Rational Players

Daniel Sgroi[a,c]
Department of Economics
University of Warwick

Daniel John Zizzo[b]
School of Economics,
University of East Anglia

Abstract

We present a neural network methodology for learning game-playing rules *in general*. Existing research suggests learning to find a Nash equilibrium in a *new* game is too difficult a task for a neural network, but says little about what it will do instead. We observe that a neural network trained to find Nash equilibria in a known subset of games will use self-taught rules developed endogenously when facing new games. These rules are close to payoff dominance and its best response. Our findings are consistent with existing experimental results, both in terms of subject's methodology and success rates.

JEL Classification: C72, D00, D83.

Keywords: neural networks, normal form games, bounded rationality.

[a] Corresponding author. Address: Department of Economics, University of Warwick, Gibbet Hill Road, Coventry CV4 7AL, UK. Email: daniel.sgroi@warwick.ac.uk. Telephone: +44 2476 575557. Fax: +44 2476 523032.

[b] Address: School of Economics, University of East Anglia, Norwich, NR4 7TJ, UK. Email: d.zizzo@uea.ac.uk.

[c] Both authors would like to thank Michael Bacharach, Vince Crawford, Huw Dixon, Glenn Ellison, Jim Engle-Warnick, Susan Hurley, Gerry Mackie, Meg Meyer, Matthew Mulford, Raimondello Orsini, Andrew Oswald, Michael Rasmussen, Antonio Santamaura, Andrew Temple and participants to presentations in Cambridge, Copenhagen and Oxford. The online technical appendices can be found at http://www.uea.ac.uk/~ec601/nnt.pdf and at http://www2.warwick.ac.uk/fac/soc/economics/staff/faculty/sgroi/papers/nnt.pdf

1 Introduction

In this paper we examine how a neural network can learn to do well in some situations and then use this training to do well in others also. In other words, we wish to model how to learn game-playing rules in general through a process of learning by example. As a metaphor, this is much like the process of training a neural network. Imagine an economic agent going through a training period, perhaps being educated at school, or learning directly from a parent, often through example. Once felt to be ready, this newly educated agent is left to fend for itself. Next consider the similarities to the process of training a neural network: it is first trained by observing a series of examples and being informed which choice to follow, it then produces an algorithm that explains why the given choice is the right course of action in these example situations, and finally, it faces a sequence of new situations in which it must decide what to do based on the algorithm it has learned from the earlier training period.

If neural networks have the potential to be good at developing game-playing rules in general, then we have an interesting general question to answer: can a neural network learn how to play any $n \times n$ game by playing a finite set of other $n \times n$ games? To give a feel for what we are trying to discover, consider how experience of chess might help to play checkers, how experience as a bank manager might help someone to be a better store manager, or how experience in an oligopoly competing in prices should surely assist an oligopoly that competes in quantities. At least some experimental evidence from signaling games (Cooper and Kagel 2003), two 2×2 and two 3×3 games (Weber 2003), a stylized job search environment (Slonim 1999), a compound lotteries choice task (Zizzo 2005) and various psychological experiments (Fantino and Stolarz-Fantino 2005), suggests that transfer of learning can be possible to some degree even within the limited time horizon of an experimental session, though limitations exist. Rather than being interested in transfer of learning between individual games in a short time span, however, our approach will be to develop and test a tentative framework to examine the long run learning of general game-playing strategies after experience has been obtained on a wide set of games.

This is a difficult task, and the methods used in this paper represent a beginning rather than the final word. In particular, the question we address is simpler: can a specific neural network trained to play well in a small set of 3×3 normal form games be expected to play well in a new set of 3×3 normal form games which it has never faced before against a population of players all of which will play Nash equilibrium strategies? 3×3 games were chosen as they represent the simplest class of $n \times n$ games which allow us to consider iterated deletion of dominated strategies. The method of investigation will be a neural network trained to play Nash equilibria coupled with its empirical testing in a set of new games. The choice of a neural network is in part justified by its relevance for transfer of learning, in part by the large body of related literature in engineering, cognitive science and cognitive psychology (e.g. the empirical evidence on linguistic learning in Rumelhart and McClelland 1986), and in part by the fact that our results are broadly

consistent with experimental tests on human subjects, for example Costa Gomes et al. (2001), Stahl and Wilson (1994 and 1995). As to the limitations of the paper, we focus on a single neural network player in a population of Nash players, not a population of neural networks. The latter would represent a different set of considerations but would be equally interesting. Second, we consider a neural network trained to select Nash equilibria, not trained to maximize utility, so we are assuming Nash equilibria to be the best way to play. This is justified in part below, but clearly our results are only applicable where Nash equilibria are generally accepted to be the correct solution method. In particular this paper can be said to address a subsidiary question: even with direct exposure to the concept of Nash equilibrium taught through example, will a neural network player stick with this solution concept or independently develop a new technique for solving new games? What we discover is that Nash equilibrium is just too complex a concept for a *loosely biologically plausible* neural network to use in general in new environments. It is not difficult for the network to find Nash equilibria in specific games, but what is difficult is to learn to employ Nash as a general algorithm on the basis of learning by example.

This work is complementary to, but very different from, evolutionary game theory and other well documented methods of studying bounded rationality as our focus is to addressing the practical concern of finding a biologically reasonable learning model which picks out Nash equilibria at a similar rate to real-world subjects. There are several pioneering papers which address the use of neural networks as models of bounded rational agents in economics. These tend to focus on how neural networks perform at specific tasks, such as repeated instances of the Prisoner's Dilemma in Cho and Sargent (1996), Cho (1995) and Macy (1996) and games of moral hazard in Cho and Sargent (1996), Cho (1994) and Cho (1996); heterogenous consumers in a model of monopoly in Rubinstein (1993); market entry with bounded rational firms in Leshno et al. (2003); Cournot oligopoly in Barr and Saraceno (2005); and finally inter-generational learning in Hutchins and Hazelhurst (1991). Sgroi (2005) provides a recent survey of this literature.

Our work differs from most reinforcement learning models (whether action-based, belief-based, or based on replicator dynamics) in trying to explain transfer of learning between games and not just within each given game.[1] An important exception is the literature on rule learning models (e.g., Stahl 2000, 2001, 2005). We do not see the neural network model in this paper as a competitor to a rule learning approach. Part of the contribution of this paper is to determine which rules emerge *endogenously* from neural network learning, whereas a rule learning model, *given* a set of rules, determines which one becomes more frequent or less frequent. Thus, there is a sense in which our model is situated one logical step prior to the rule learning model.[2]

Automata and finite state machines share similarities with neural networks, and both face

[1] See Stahl (2005) for a good overview.

[2] In addition, the focus of this paper is not on learning during an experiment (i.e., the fit of data related to learning in the context of an experimental session) but rather on the fit between Nash and other algorithms and what the neural network has learnt in long run computer simulations.

similar difficulties learning Nash strategies.[3] For example Gilboa and Zemel (1989) show that computing the Nash equilibrium of a one-shot normal form game is NP-hard, and Gilboa (1988) and Ben-Porath (1990) characterize the complexity problems of computing best response strategies in repeated games. The closest papers to ours in the automata literature are by Miller (1996) and Ho (1996), who show how automata can be used to model strategy learning in repeated games. However our work differs in two important ways. First and most importantly, we are concerned with learning game-playing rules in general for new games, not for the same game faced in a repeated context. Second, neural networks utilize parallel processing rather than serial processing and thus learn through a different mechanism (see MacLeod et al. 1998).

To summarize, this paper first provides a summary of the literature on the theoretical limitations to neural network learning. Next it argues that these limitations are *precisely the sort of limitations we want to see when modeling bounded rationality* since they evolve endogenously and reflect a method of learning that is loosely based on biological plausibility. We examine some statistical results derived from neural network learning, and we show that these are consistent with existing experimental evidence on human cognitive abilities. The net result is an imperfect model of learning which provides suggestive insights into observed imperfect human learning.

1.1 Overview

The next section details the model to be used, in particular defining the neural network player and the games it is to face. It also details some existing results in neural network and algorithm complexity theory which provide us with clear theoretical predictions about what to expect our neural network player to be able to achieve. This section also provides an extended literature review section for those with limited exposure to neural networks; however, those with a thorough grounding in neural network theory or those who wish to focus on results may wish to skip part of section 2, or go straight to section 3 which details the results of the empirical testing of the network. Section 4 concludes.[4]

2 A Primer on Neural Networks

Dealing with a neural network as a model of bounded rational behavior presents us with a problem. While neural networks are known and used within economics, they are mainly used as an econometric tool, not as a behavioral model. Therefore, this section presents a primer on neural networks together with an extensive survey of related results. For those with no

[3] We are grateful to an anonymous referee for pointing out that there is a one-to-one relationship between a type of neural network and a type of finite state machine, as this leaves the door open for the techniques introduced in this paper to be applied to repeated games in future research.

[4] A technical appendix which includes more detail on backpropagation, results on convergence, and definitions of the solution concepts in section 3 is available online from the authors' websites.

exposure to neural networks, section 2 can be supplemented with material in relevant texts, such as Anthony and Bartlett (1999), White (1992) or Sgroi (2005).

To summarize, existing work on the learning problem faced by a neural network suggests that a neural network player cannot be expected to find the Nash equilibria consistently in completely new games, even though a neural network player does build a methodology for such games. However, the method used by a neural network will have certain characteristics which will enable it to be successful in a subset of cases, but not all. While we can of course improve its performance to the point of virtual perfection, to do so would require the addition of biologically implausible numerical techniques and would rob the neural network model of any claim to model bounded rational play in human subjects.[5] The material in section 2 sets the stage for examining whether the theorized characteristics of a neural network player get close to the limitations of observed human play, which provides the focus for the estimation in section 3.

2.1 Defining a Neural Network

Neural networks can be loosely described as *artificial intelligence models inspired by analogy with the brain and realizable in computer programs.* They typically learn by exposure to a series of examples (a training set) and adjustment of the strengths of the connections between its nodes. They are then able to do well not only on the original training set, but also when facing problems never encountered before. Consider a neural network C to be a machine capable of taking on a number of states, each representing some computable functions mapping from input space to output space, with two *hidden* layers of further computation between input and output. Hidden layers can be thought of as intermediate layers of computation between input and output. Since we see the input go in and the output come out, but do not directly see the activity of intermediate layers, they are described as hidden.

Definition 1 *Define a neural network as* $C = \langle \Omega, X, Y, F \rangle$ *where* Ω *is a finite set of states,* $X \subseteq \mathbb{R}^n$ *is a set of inputs,* Y *is a set of outputs and* $F : \Omega \times X \mapsto Y$ *is a parameterized function. For any* ω *the function represented by state* ω *is* $h_\omega : X \mapsto Y$ *given by* $h_\omega(x) = F(\omega, x)$ *for an input* $x \in X$*. The set of functions computable by* C *is* $\{h_\omega : \omega \in \Omega\}$*, and this is denoted by* H_C*.*

Put simply, when the network, C, is in state $\omega \in \Omega$, it computes the function h_ω providing it is computable. The state ω is a reduced form expression encapsulating past experience and updating by the neural network, leading to a choice of function h_ω. The parameterized function F is also reduced form, capturing the hidden layers. In order to produce answers which reasonably

[5] The potential ability of neural networks to perform almost flawlessly is discussed towards the end of section 2.4. As noted there, supplementing a neural network's learning algorithm with a guessing stage, such as grid search or an application of the theory of sieves, could enable the neural network to perform much better; however this would lack biological plausibility and be extremely processor hungry. White (1992) goes into considerable detail on this point.

correspond to a notion of correctness (in this case we will restrict this to be the unique Nash strategy in a 3×3 game), we need to *train* the network. Let us start by defining an activation function.

Definition 2 *An activation function for node i of layer k in the neural network C is of the logistic (sigmoid) form $\theta_i^k = \left[1 + \exp\left(-\sum_j w_{ij}^k u_{ij}^{k-1}\right)\right]^{-1}$ where u_{ij}^{k-1} is the output of node j in layer $k-1$ sent to node i in layer k (hence forming the input to layer k), and w_{ij}^k is the weight attached to this by node i in layer k. The expression $t_i \equiv \sum_j w_{ij}^k u_{ij}^{k-1}$ is the total activation flowing into node i.*

Finally, we need to specify the situation faced by the neural network. For this we consider a normal form game $G = \langle N, \{A_i, \pi_i\}_{i \in N} \rangle$ of perfect information with a unique pure strategy Nash equilibria. Actions are given by $a_i \in A_i$. Feasible action combinations are given by $A = A_1 \times A_2$. Payoffs for player i are given by $\pi_i : A \mapsto \mathbb{R}$. We normalize payoffs to be drawn from a uniform distribution with support $[0, 1]$ which are revealed to the players before they select an action.

2.2 Training

Consider a set of 18 input nodes each recording and producing as output a different value from the vector $x_k = (x_k^1, ..., x_k^{18})$. This neatly corresponds to the payoffs of a 3×3 game. Now consider a second set of 36 nodes (the first hidden layer). Each node in this second layer receives as an input the sum of the output of all 18 input nodes transformed by the activation function of node i in layer 2. All of the nodes in the second layer send this output θ_i^2 to all nodes in the second hidden layer, which weights the inputs from all i of the first hidden layer by the activation function to produce θ_i^3. These numbers are sent to the final layer of two nodes to produce an output y which forms a 2-dimensional vector representing the choice of strategy in a 3×3 game. To explain this representation of a strategy in a 3×3 game for the row player, the vector $(1, 0)$ would imply that the neural network player's choice is the pure strategy embodied by selecting the top row, $(0, 1)$ would imply the middle row, and $(0, 0)$ the bottom row. Of course there is nothing restricting the neural network from choosing values other than 0 or 1, so it might select $(0.8, 0.2)$, which would suggest that it is certain it does not wish to pick the bottom row strategy, and is fairly happy to pick the top row strategy, but still has some doubts about whether it is better than middle. Should the Nash equilibrium be $(1, 0)$ we would aim to train the network to get close to $(1, 0)$ in a sense to be defined below. The network's outputs will be interpretable as a probability vector only as long as their sum adds up to 1, so for example $(0.9, 0.9)$ is ruled out, and normalized to $(0.5, 0.5)$.

Training essentially revolves around finding the set of weights that is most likely to reproduce the actual Nash equilibrium of the game faced. During training C receives a sequence of M

random games called a *training sample*:

$$x^M = \left(\left(x_1^1, ..., x_1^{18}\right), \left(x_2^1, ..., x_2^{18}\right), ..., \left(x_M^1, ..., x_M^{18}\right)\right) = (x_1, x_2, ..., x_M) \in X^M$$

until some stopping rule determines the end of the training at some round T. If $T > M$, then (some or all of) the random games in M will be presented more than once. The labelled examples x_i are drawn independently according to the uniform $[0, 1]$ probability distribution P_T which represent the payoffs of a 3×3 game, subject to the condition that each vector x_t ensures the existence a unique Nash equilibrium in pure strategies. If this condition fails a new vector is drawn from P_T. A random training sample of length M is an element of X^M distributed according to the product probability distribution P^M.

Assume that $T > M$. In this case, training might be *sequential*: after $q \times M$ rounds (for any positive integer q s.t. $q \times M < T$), M is presented again, exactly in the same order of games. If training is *random without replacement*, it is less restricted to the extent that the order in which the random games are presented each time is itself random. If training is *random with replacement*, in each round the network is assigned randomly one of the random games in M, until round T. Having selected a sample sequence of inputs, x, and determined the unique Nash strategy associated with each, α, we need to consider how C *learns* the relationship between the two to ensure that its output y will approach the Nash strategy.

Definition 3 *Define the network's root mean square error ε as the root mean square difference between the output y and the correct answer α over the full set of $q \times M$ games where individual games are indexed by i, so our error function is $\varepsilon \equiv \frac{1}{q \times M}[\sum_{i=1}^{q \times M} (y_i - \alpha_i)^2]^{1/2}$.*

Note that the unique nature of a pure strategy means that the mean square error need only be one dimensional. For example a pure strategy Nash equilibrium of $(1, 0)$ as compared with an output of $(0.9, 01)$ allows the difference $1 - 0.9$ to form the basis of the means square error. The aim is to minimize the error function by altering the set of weights w_{ij} of the connections between a typical node j (the sender) and node i (the receiver) in different layers. These weights can be adjusted to raise or lower the importance attached to certain inputs in the activation function of a particular node. The correct answer here is the vector associated with the unique Nash equilibrium in pure strategies. In principle we could use any other measure, including for example training the neural network to select the best or even worst outcome in terms of game payoff. This paper's focus however is narrowly limited to a study of a neural networks' ability to learn to pick the unique pure strategy Nash equilibrium in G.

2.3 Backpropagation

Generally, the optimum parameter or set of parameters designed to minimize the error function cannot be calculated analytically when a model is nonlinear, so we must rely on a form of

numerical optimization. The method we use is called backpropagation, specified in Rumelhart et al. (1986), and is the most standard method used in the neural network literature for building practical neural networks. The basic intuition behind backpropagation is that of psychological reinforcement: the economic decision-maker tries to learn how to perform better in the task, and the more disappointing the outcome (relative to the 'correct' outcome), the deeper the change in connection weights will be. Unlike the reinforcement learning or belief-based learning models of, for example, Roth and Erev (1995 and 1998) or Camerer and Ho (1999), reinforcement learning under backpropagation does not occur directly over actions or beliefs but rather over connection weights: the parallel processing by the network when a new stimulus is received will be a function not just of the connection weights and the stimulus received but also of the network architecture.

Backpropagation requires a teacher explicitly telling the correct answer during training, and this might appear too strong a requirement: it renders backpropagation a *more powerful* algorithm than is biologically plausible. Backpropagation is more powerful also in another sense: it adjusts individual connection weights using *global* information on how to best allocate output error which is unlikely to occur in biological brains as discussed in MacLeod et al. These limitations, however, should not be overstated: what they suggest is that backpropagation might be a plausible *upper bound* to the learning of biological neural networks of some given size. Conversely, stronger learning algorithms, of the kind used in White to show learnability, are much further from biological or cognitive plausibility. Hence, the non-learnability results with backpropagation discussed in the next section cannot be easily dismissed as an artificial product of too weak a learning rule.[6]

To give an overview of the backpropagation method, we first compute the error of the output layer (layer N) and update the weights of the connections between layer N and $N-1$.[7] We then compute the error to be assigned to each node of layer $N-1$ as a function of the sum of the errors of the nodes of layer N that it activates. We follow this procedure backwards iteratively, one layer at a time, until we get to layer 1, the input layer. Key parameters include a *learning rate* given by $\eta \in (0, 1]$, a parameter of the learning algorithm which must not be chosen to be too small or learning will be particularly vulnerable to local error minima, and *momentum* $\mu \in [0, 1)$ which makes connection changes smoother by introducing positive autocorrelation in the adjustment of connection weights in consecutive periods. The connection weights of the network are updated using backpropagation until round T. T itself can be determined exogenously by the builder of the neural network, or it can be determined endogenously by the training process (i.e. training stops when the network returns the correct output with ε lower than a target value).

[6] In practice we know that a neurotransmitter, dopamine, plays a role in biological neural networks analogous to that of the teacher in the backpropagation algorithm; the activation level of dopamine neurons may work as a *behavioral adaptive critic* (i.e. it tells the agent how to adapt its behavior to successfully deal with a task). Zizzo (2002) provides more detail on this.

[7] More detail on backpropagation is available in a supporting technical appendix available from the authors, and in Rumelhart et al.

To summarize, a neural network is shown a set of games with a unique Nash equilibrium and computes an algorithm which characterizes the relationship between each game and the unique Nash strategy. It continues to do so until it is found to be able to recognize the Nash equilibrium in each of these "training" games with mean squared error, ε, below a certain threshold, where ε measures the distance of the network's output from a vector representation of the pure Nash equilibrium. At this point the network is pronounced "trained" and allowed to use the algorithm it has developed to search for the unique Nash equilibrium in a series of new games that were not part of the training sample. This is widely known in the neural network literature as supervised learning and accords with an intuitive notion of a teacher continuously correcting the behavior of a student until behavior is close to that expected in a Nash equilibrium. When it has achieved this or close enough to it (when it knows the best way to play in this set of games), it is shown some different games and asked to find the Nash equilibria for these without ever having seen these new games before. It can however use the algorithms (rules) it has already learned, in order to allow it to choose correctly the Nash equilibria from those games which it has seen before (the training set).

2.4 Learnability Results in the Literature

Many results exist in the algorithm complexity and computer science literature which stress the difficulty of the learning problem. One of the most well-known results comes from Hornik et al. (1989): there exists a set of weights for a standard *feedforward* network with only a single hidden layer which allow it to approximate any continuous function uniformly on any compact set and any measurable function arbitrarily well. However, the network may experience inadequate learning so that the learning dynamic will fail to reach the global error-minimizing algorithm. A *learning algorithm* L takes random training samples and acts on these to produce a function $h_\omega \in H$ that, provided the sample is large enough, is with probability at least $1-\delta$, ε-good (with ε defined as in definition 3) for P_T. It can do this for each choice of ε, δ and P_T. Closely related is the definition of learnability.[8]

Definition 4 *A learning algorithm L takes random training samples and acts on these to produce a hypothesis $h \in H$. We say that the class of functions H is learnable if $\exists$ a learning algorithm L for H.*

Thus we see how crucial is the computability of our function h_ω (h for a given state ω) which represents the entire processing of the neural network's multiple layers, taking an input vector x and producing a vector representation of a choice of strategy. Over a long enough time period we would hope that C will return a set of optimal weights which will in turn produce

[8] A more formal set of definitions "learnable" and "learning algorithm" can be found in Sgroi and Zizzo (2002).

a function which will select the Nash strategy if there exists a learning algorithm for selecting Nash equilibria (H in this case). Alternatively if we wish to attain some below perfect success rate, we can do so using a finite training sample, and the success rate will grow as the number of examples increases. This all crucially rests on the ability of backpropagation to pick out the globally error-minimizing algorithm for finding Nash equilibria. Let us now define C's learning problem.

Definition 5 *C, using backpropagation, faces a training sample of size $M \times q$. The Nash problem is to find an algorithm for which $\varepsilon \to 0$ as $M \to \infty$ where the error function ε is as defined in definition 3.*

Sontag and Sussmann (1989) demonstrates that backpropagation converges only to a local minimum of the error function.[9] The problem is exacerbated in the case of our neural network C as the space of possible weights is so large. Furthermore, Fukumizu and Amari (2000) shows that local minima will always exist in problems of this type, and Auer et al. (1996) shows that the number of local minima for this class of networks can be exponentially large in the number of network parameters. The upper bound for the number of such local minima is calculable, but it is unfortunately not tight enough to lessen the problem (see Sontag 1995). In fact, as the probability of finding the absolute minimizing algorithm (the Nash algorithm) is likely to be exponentially small, the learning problem faced by C falls into the NP-hard class of problems.[10]

A gradient descent algorithm such as backpropagation cannot consistently find an absolute minimum of the error function given the prevalence of local minima. Several statements of this exist within the algorithm complexity literature. For instance, in Anthony and Bartlett (1999), Theorem 25.5 states that problems of the type given in definition 5 faced by the class of networks encompassing C are NP-hard. There exist several forms of this result for different types of network including the feedforward class of which C is a member. As an example, we shall show later that the trained network performs well in games A and B in Table 5 but poorly in games C and D, and we shall discuss why this is the case.

Other far less biologically plausible methods involving processor hungry guess and verify techniques can produce better results. If we were to supplement the algorithm with a guessing stage (i.e. add something akin to grid search or a subtle application of the theory of sieves), then we could hope to find the absolute minimum in polynomial time however, White (p. 161) argues

[9] White summarizes the difficulties inherent in backpropagation: it can get stuck at local minima or saddle points, can diverge, and therefore cannot be guaranteed to get close to a global minimum. In fact, while "sufficiently complex multilayer feedforward networks are capable of arbitrarily accurate approximations to arbitrary mappings...an unresolved issue is that of "learnability", that is whether there exist methods allowing the network weights corresponding to these approximations to be learned from empirical observation of such mappings (p.160)."

[10] For more on NP-hardness, see the small related literature on the complexity of computing an automaton to play best response strategies in repeated games, for example Ben-Porath (1990) and Gilboa (1988).

that such methods "... lay no claim to biological or cognitive plausibility" and are therefore not desirable additions to the modeling of decision-making. For this reason we will restrict attention to backpropagation, and so we cannot consider the task facing the network to be *learnable* in the sense of definition 4.

The problem of NP-hardness is acute; the solution can be found in exponential time, but not in polynomial time. For any network with a non-trivial number of parameters, such as C, the difference is great enough for us to consider the problem intractable: backpropagation cannot consistently find an algorithm capable of providing Nash equilibria in never before seen games.

To summarize, the neural network player will find a decision-making algorithm that will retain some error even at the limit, so we may have to be content with an algorithm which is effective in only a subclass of games, optimizing network parameters only in a small subspace of the total space of parameters. In the case of normal-form games we can summarize what can be extracted from the existing literature for our particular problem as follows: *with high probability our player will not learn the globally error-minimizing algorithm for selecting Nash equilibria in normal-form games.* However, we can reasonably assume that some method will be learned, and this should at least minimize error in some subset of games corresponding to the domain of some local error-minimizing algorithm.

2.5 Local Error-Minimizing Algorithms

We are left with the question, what is the best our neural network player can hope to achieve? If we believe the neural network with a large, but finite training set adequately models bounded-rational economic agents, but cannot flawlessly select Nash strategies with no prior experience of the exact game to be considered, this question becomes: what is the best a bounded-rational agent can hope to achieve when faced with a population of fully rational agents? In terms of players in a game, we have what looks like bounded-rational learning or satisficing behavior: the player will learn until satisfied that he will choose a Nash equilibrium strategy sufficiently many times to ensure a high payoff. We label the outcome of this bounded-rational learning as a local error-minimizing algorithm (LMA). More formally, consider the learning algorithm L, and the 'gap' between perfect and actual learning, $\varepsilon_0(M,\delta)$. Z^M defines the space of possible games as perceived by the neural network.

Definition 6 *If* $\exists$ *a* $\varepsilon_0(M,\delta)$ *s.t.* $\forall_{M,\delta,P_T}$, *with probability at least* $1-\delta$ *over all* $z \in Z^M$ *chosen according to* P^M, $er_p(L(z)) < opt_p(H) + \varepsilon_0(M,\delta)$, *and* $\forall_{\delta\in(0,1)}$, $\varepsilon_0(M,\delta) \to 0$ *as* $M \to \infty$ *then* $\varepsilon_0(M,\delta)$ *is defined the global error-minimizing algorithm (GMA).*

This states that for all possible games faced by the network, after sufficient training, the function will get arbitrarily close to the Nash algorithm, collapsing the difference to zero. This requires an algorithm sufficiently close to Nash to pick a Nash equilibrium strategy in almost all games.

Definition 7 *A local error-minimizing algorithm (LMA) will select the same outcome as a GMA for some $z \in Z^M$, but will fail to do so for all $z \in Z^M$.*

LMAs can be interpreted as examples of rules of thumb that a bounded-rational agent is likely to employ in the spirit of Simon (1955 or 1959). They differ from traditionally conceived rules of thumb in two ways. First, they do select the best choice in some subset of games likely to be faced by the learner. Second, they are learned *endogenously* by the learner in an attempt to maximize the probability of selecting the best outcome where the 'best' outcome can be determined in terms of utility maximization or a reference point, such as the Nash equilibrium.

3 Testing a Neural Network Model

We have seen that when we restrict the learning algorithm employed by a neural network to be backpropagation, generally thought to be if anything too strong an algorithm in practice, we find that the neural network will not be able to pick Nash equilibria in new games faultlessly. While we could improve its performance easily through the addition of other algorithms, we instead restrict our attention to probably the most biologically plausible algorithm and ask whether such a limited neural network approximates observed human failings, and if so, whether the methods it employs are a good model of bounded rationality.

3.1 Setting the Scene

In practical terms we can construct a simulated network and test to see whether this network matches our theoretical predictions. The training set is a sequence of inputs $x \in X$ corresponding to the set of actions A_i for N players in M random games, and outputs corresponding to the payoffs $\pi_i : A \mapsto \mathbb{R}$ for N players for each of the actions. We set $M = 2000$, $N = 2$ and restrict the action set by assuming a 3×3 normal-form game. 2×2, 2×3 and 3×2 games could be modeled by forcing rows or columns of zero. We then allow the network to play 2000 further random games never encountered before, selecting a single input and recording a single output. Since we force each game to contain a unique Nash equilibrium in pure strategies and we restrict the network's choice to be in pure strategies, we can then check the network's *success rate* as defined by the proportion of times the network selected the Nash strategy to within a given threshold of mean squared error (as defined in definition 3). For example, if the correct output is $(1, 0)$ and the neural network returns $(0.99, 0)$, it easily meets an $\varepsilon = 0.05$ threshold.

The training set consisted of $M = 2000$ games with unique pure Nash equilibria. Training was random with replacement (subject to the unique Nash equilibrium condition) and continued until the error ε converged below 0.1, 0.05 and 0.02 (i.e. three *convergence levels* γ were used:

more than one convergence level was used for the sake of performance comparison).[11] Convergence was checked every 100 games, a number large enough to minimize the chance of a too early end of the training; clearly, even an untrained or poorly trained network will get an occasional game right, purely by chance. The computer determined initial connection weights and order of presentation of the games according to some 'random seed' given at the start of the training. To check the robustness of the analysis, C was trained 360 times, that is once for every combination of 3 learning rates η (0.1, 0.3, 0.5), 4 momentum rates μ (0, 0.3, 0.6 and 0.9) and 30 (randomly generated) random seeds. Momentum rates span the whole range, learning rates reflect a plausible range when backpropagation is used, and 30 random seeds were chosen as a number large enough to avoid dependence on specific values or idiosyncratic combinations of values. Convergence was always obtained, at least at the 0.1 level, except for a very high momentum rate.[12] We will henceforth call the simulated network C^* once trained to a given convergence level.

C^* was tested on a set of 2000 games with unique Nash equilibria never encountered before.[13] We restricted ourselves to training with games with unique pure strategy Nash equilibria because of the technical need for backpropagation to be able to work with a unique solution. This appeared a lesser evil relative to having to postulate refinements to Nash in order to discriminate among multiple equilibria and, hence, making the analysis dependent on these refinements. We considered an output value to be correct when it is within some range from the exact correct value. If both outputs are within the admissible range, then the answer can be considered correct; see Reilly (1995). The ranges considered were 0.05, 0.25 and 0.5, in decreasing order of precision.

[Table 1: Percentage of Correct Answers]

Table 1 displays the average performance of C^* classified by γ, η and μ.[14] It shows that C^* trained until $\gamma = 0.1$ played exactly (i.e., within the 0.05 range) the Nash equilibria of 60.03%

[11] It is important that the sample the network is trained on is sufficiently large and representative. If shown a small non-random selection of games, the network will not be trained effectively. For example, just showing the games used by Costa Gomes et al. in their experiment would not do for training and might result in overfitting (ignoring for the sake of the argument the issue of dimensionality, as Costa Gomes et al. do not just have 3×3 games).

[12] More detail on convergence is available in a supporting technical appendix available online.

[13] Sgroi and Zizzo (2002) and Zizzo and Sgroi (2000) consider how C^* performs when faced with games with multiple Nash equilibria.

[14] The level of convergence γ simply measures how correct we ask the network to be: the smaller it is, the stricter the criterion. The learning rate η is a coefficient that determines the speed of the adjustment of the connection weights when the network fails to play the Nash equilibrium behavior. A positive momentum rate μ introduces autocorrelation in the adjustments of the connection weights when successive examples are presented. The error tolerance criterion measures how close the answer given by the network must be to the exact answer in order to consider the answer right. The smaller the error tolerance criterion, the tighter it is. The numbers given under 'At least 1 Correct Output' are the % of cases in which at least 1 of the two output nodes is correct. The numbers given under 'Correct Answer Given' are the % of cases in which both output nodes are correct.

of the testing set games (e.g. of 2000 3×3 games never encountered before). This fits well with the 59.6% average success rate of human subjects newly facing 3×3 games in Stahl and Wilson (1994), although one has to acknowledge that the sample of games they used was far from random. With an error tolerance of 0.25 and 0.5, the correct answers increased to 73.47 and 80%, respectively. Further training improves its performance on exactness (the 0.02-converged C^* plays the Nash equilibria of a mean 2/3 of games) but *not* on 'rough correctness' (the 20% result appears robust). This suggests (and indeed further training of the network confirms) that there is an upper bound on the performance of the network. Table 1 also shows that once C converges, the degree it makes optimal choices is not affected by the combination of parameters used: the average variability in performance across different learning rates is always less than 1%, and less than 2% across different momentum rates. This is an important sign of robustness of the analysis.

We compared C^*'s performance with three null hypotheses of zero rationality. Null1 is the performance of the entirely untrained C: it checks whether any substantial bias towards finding the right solution was hardwired in the network. Null2 alternates among the three pure strategies: if C^*'s performance is comparable to Null2, it means all it has learned is to be decisive on its choice among the three. Null3 entails a uniformly distributed random choice between 0 and 1 for each output: as such, it is a proxy for zero rationality. Table 2 compares the average performance of C^* with that of the three nulls.[15] Formal t tests for the equality of means between the values of C^* and of each of the nulls (including Null2) are always significant ($p < 0.0005$). C^*'s partial learning success is underscored by the fact, apparent from Tables 1 and 2, that when C^* correctly activates an output node, it is very likely to categorize the other one correctly, while this is not the case for the nulls.

[Table 2: Average Performance of the Trained Network versus Three Null Hypotheses]

3.2 Is the Neural Network using Alternatives to Nash?

It appears that C^* has learned to generalize from the examples and to play Nash strategies at a success rate that is significantly above chance. Since it is also significantly below 100%, the next question we must address is how to characterize the LMA achieved by the trained network. While the network has been trained to recognize Nash equilibria when they are uniquely defined, from section 2 we know that the process of calculating a Nash equilibrium is a hard one. Our first strategy is therefore to ask ourselves whether there are simple alternatives to Nash capable of describing what the network does better than Nash on the games over which they are uniquely

[15] The smaller the error tolerance criterion, the tighter the criterion used to consider C^*'s strategy choice correct.

defined. Given the robustness of our analysis in the previous section to different combinations of η and μ, in this and the next sections we focus just on the case with $\eta = 0.5$ and $\mu = 0$. The robustness of the results with different parameter combinations ensures that *this particular choice* is not really relevant. In any event, it was driven by two considerations: 1. any momentum greater than 0 has hardly any real psychological justification, at least in this context, and 2. given $\mu = 0$, a learning rate of 0.5 had systematically produced the quickest convergence.

For testing we used the 30 networks trained with the 30 different random seeds but with the same learning (0.5) and momentum (0) rates. Using these 30 networks, we tested the average performance of the various algorithms on the same testing set of 2000 new games with unique pure Nash equilibria considered in the previous section.

We consider the following algorithms or alternative solution methods in turn: 1) minmax, 2) rationalizability, 3) '0-level strict dominance' (0SD), 4) '1-level strict dominance' (1SD), 5) 'pure sum of payoff dominance' (L1), 6) 'best response to pure sum of payoff dominance' (L2), 7) 'maximum payoff dominance' (MPD), and 8) 'nearest neighbor' (NNG). 1 and 2 are well known, and 3 and 4 are simply levels of reasoning in the rationalizability process (rationalizability equating to '2-level strict dominance' in a 3×3 game).[16] Intuitively, MPD corresponds to going for the highest conceivable payoff for itself, L1 to choosing the best action against a uniformly randomizing opponent, and L2 to choosing the best action against a L1 player. Finally, a NNG player responds to new situations by comparing them to the nearest example encountered in the past and behaves accordingly. These algorithms seem worth testing as plausible solution methods partly because they are among the most well-known methods of solving a game that are simpler than Nash and partly because they accord to different possible heuristics which might tempt a player, such as going for large numbers (MPD or L1; for the latter, see Costa Gomes et al.), responding to someone going for large numbers (L2; see Costa Gomes et al.) or using similar experiences from the past (NNG; see Gilboa and Schmeidler 2001).

We define a game as *answerable* by an algorithm if a unique solution exists. Table 3 lists the number and percentage of answerable games (out of 2000) according to each algorithm, averaged out across the 30 neural networks trained with different random seeds, $\eta = 0.5$ and $\mu = 0$.

[Table 3: Answerable Games and Relationship to Nash]

Table 3 also lists the percentage of games where the unique solution coincides with the pure Nash equilibrium of the game. In order to determine how an agent following a non-Nash algorithm would behave when faced with the testing set, we need to make an auxiliary assumption with regards to how the agent would be playing in non-answerable games. We assume that, in non-answerable games, the agent randomizes over all the actions (two or three, according to the

[16] More detail about these algorithms is available in a supporting technical appendix available online.

game) admissible according to the non-Nash algorithm (e.g., in the case of rationalizability, all the non-dominated actions); if the admissible actions are two or three and one of them is the Nash equilibrium choice, the agent will get it right 1/2 or 1/3 of the times on average, respectively. The right column of Table 3 adjusts accordingly the expected success rate of the non-Nash algorithm in predicting Nash, giving us the degree to which the various algorithms are good or bad LMAs.

[Table 4: Describability of C^*'s Behavior by Non-Nash Algorithms]

Some findings emerge. Our set of candidate LMAs typically can do better than how a zero rational agent simply playing randomly across all choices and games would do. More strategically sophisticated LMAs can do better than less strategically sophisticated ones. Rationalizability, 0SD and 1SD are limited in their ability in predicting Nash by the limited number of corresponding answerable games. The most successful algorithms in predicting Nash are first L2, then rationalizability and finally L1. L2 and L1 combine, in different proportions, simple algorithms based on payoff dominance with considerable Nash predictive success in our set of 3×3 games. They have also been found as the best predictors in normal-form games of behavior by experimental subjects in Costa Gomes et al. L2 is particularly impressive in predicting Nash in our set of 3×3 games. On the basis of these considerations, we hypothesize that the LMA played by C^* may be describable to a significant degree by L2 and also possibly L1, among the non-Nash algorithms we have considered. In our interpretation, though, we do not rule out the possibility that C^* does more than simply follow any of the non-Nash algorithms of Table 3. Still, if true, it would be consistent with the predictive success of L2 and L1 in experimental data in Costa Gomes et al., even though they did not include 3×3 games in their experiment. Table 4 shows how well the various algorithms can describe C^*'s behavior on the testing set. We consider both the success rate as a percentage of the answerable games or of the full testing set and an adjusted success rate to allow once again for random play over admissible strategies in the non-answerable games.

NNG fares considerably worse than Nash on the data; indeed, it does worse in predicting C^*'s behavior than it does in predicting Nash (see Table 3). We should not be surprised by the fact that the NNG still gets about half of the games right according to the 0.02 convergence level criterion; it is quite likely that similar games will often have the same Nash equilibrium. Partial nearest neighbor effects cannot be excluded in principle on the basis of Table 4. However, the failure of the NNG algorithm relative to Nash suggests that, at least with a training set as large as the one used in the simulations ($M = 2000$), the network does not reason simply working on the basis of past examples.

Rationalizability, 0SD and 1SD outperform Nash for the games they can solve in a unique way. 0SD, 1SD and rationalizability exactly predict C^*'s behavior in 80.98%, 76.25% and 74.36%

of their answerable games, respectively; this is 8-14% above Nash. The fact that C^* still gets three quarters of all rationalizable games exactly right suggests that it does behave as if capable of some strategic thinking. However, the network can still play reasonably well in games not answerable according to 0SD, 1SD and rationalizability; hence, Nash still outperforms over the full testing set.

To summarize, L2 is the best algorithm in describing C^*'s behavior, with a performance comparable to rationalizability, 0SD and 1SD for answerable games but, unlike those, with virtually all the games answerable. It predicts C^*'s behavior exactly 76.44% over the full testing set. L1 performs worse than L2, but its performance still matches rationalizability over the full testing set. The fact that L2 outperforms L1, while being more strategically sophisticated, confirms that C^* behaves as if capable of some strategic thinking.

We can now turn back to the issue of which games are such that C^* performs better and which games are such that it perform worse. Where C^*'s LMA's predictions coincide with Nash, we expect C^* to be more likely to reach the right Nash answer. The median (mean) root mean square error ε when Nash coincides with L1 and L2 is only 0.018 (0.084), but shoots up to 0.287 (0.340) for the 227 games where Nash coincides with L2 but differs from L1, and to as much as 0.453 (0.461) for the subset of 103 games where Nash differs from both L1 and L2.

[Table 5: Examples of C^*'s Performance, in Terms of Root Mean Square Error Levels ε, in Four Games from the Training Set]

Table 5 contains examples of games from the training set with their corresponding ε levels when faced by C^*. Game A has the same prediction for Nash, L1 and L2 (action 3), and so has game B (action 2); both have low errors, though game B displays more trembling as good payoff values (such as 0.998) exist for alternative actions. Game C predicts action 3 for Nash and action 2 for L1 and L2; Game D predicts action 2 for Nash, but action 1 for L1 and L2. Both have poor performance as C^* tends to play the L1 and L2 action, and as action 2 is especially poor from an L1 and L2 viewpoint in game D, C^* systematically fails to reach Nash in this game. The explanatory power of different LMAs in mimicking what C^* does, and further analysis of the game features that facilitate or hinder C^*'s performance, is contained in Zizzo and Sgroi, where multivariate regression analysis is employed.

An additional interesting exercise would be to pit C^* against L1, L2 or Nash to see how well it does in terms of payoffs. This could give preliminary insights on whether an evolutionary process would see C^* under-matched by its closest LMA equivalents (L1 and L2) or by Nash. While we feel that this question is best left for future research where neural networks are embedded in a proper evolutionary dynamic, our suggestive findings are that, on the testing set, C^* seems to

do as well as or just slightly better than Nash or L2 (by 2-3%), and quite better (by 6 to 9%) than L1 (see electronic appendix D for some details).[17]

4 Conclusions

This paper presented a neural network model designed to capture the endogenous emergence of bounded-rational behavior in normal-form games. Potentially any finite normal-form could be modelled in this way, though we have concentrated on 3×3 games, the simplest class of $n \times n$ games that can be subjected to iterated deletion. A neural network player, having seen a sufficiently large sample of example games in which the Nash outcome was highlighted, *could* potentially learn the Nash algorithm. However, this is highly unlikely because of the complexity of the Nash problem; effectively, the Nash algorithm is intractable by a network that uses learning algorithms, such as backpropagation, with some biological and cognitive plausibility. Hence, the network is much more likely to find some simpler way to solve the problem that allows it to get sufficiently close in a large enough number of cases to leave the network satisfied that it has found a suitable way of playing new games. This local error-minimizing algorithm would allow the network to achieve a 'satisficing' level of success in finding a Nash equilibrium in a never-before-seen game, though it would not achieve 100% success. It would correspond to one or more behavioral heuristics endogenously learned by the bounded-rational agent. This paper argues that this limited performance by a neural network is a good model of observed bounded rationality, first because it retains a level of biological plausibility, second because the methods used emerge endogenously (rather than imposed in an *ad hoc* fashion), and finally because the level of success achieved by the neural network closely resembles the results observed in laboratory experiments.

The simulation results suggest a figure of around 60% success on games never encountered before, as compared with the 33% random success benchmark. It is also broadly consistent with the 59.6% experimental figure from Stahl and Wilson (1994). Such simulations also indicate that solution concepts other than Nash get closer to explaining the simulated network's actual behavior: pure sum of payoff dominance and the best response to this strategy. These strategies, under the respective names of L1 and L2, are those most observed in the laboratory with normal-form games in the study by Costa Gomes et al. This correspondence is the more interesting because Costa Gomes et al. uses game matrices of different dimensionality from 3×3 (namely,

[17] Let us call a choice 'decided' if both network outputs are within 0.25 of a pure strategy value. Let us then assume that, for each game, C^* chooses the action which is most frequently 'decided' in the computer simulations. Call this implementation of C^* C1. We can also add the filter that we require C^* to 'decide' an action over 1/2 of the times or over 2/3 of the times in order for it to be considered C^*'s action; call these implementations of C^* as C2 and C3 respectively. Our findings are that C1 obtains an average payoff which is 0, 1 and 6% above that of Nash, L2 and L1, respectively; C2 obtains an average payoff 1, 2 and 8% above that of Nash, L2 and L1, respectively; and C3 obtains an average payoff 2, 3 and 9% above that of Nash, L2 and L1, respectively.

2×2, 2×3, 3×2, 4×2, and 2×4), suggesting that our reliance on 3×3 games is not seriously restrictive in practice. Further, in our data L2 performs better than L1, possibly because it is a considerably more successful theoretical tool in predicting Nash while being computationally only moderately more demanding.

A neural network cannot learn to pick out Nash equilibria faultlessly in a series of new games, even when it is capable of doing so in a finite subset of games. A grand master chess player may be a tough chess opponent and will be a strong player when facing many new games, but there will equally be many times when he will play new games, make many errors, and face defeat. In the process of trying to learn always to pick out Nash equilibria, a neural network will stumble onto an alternative simpler set of rules which may look at first sight like some form of simple dominance. Indeed perhaps the most interesting finding in this paper is that a simulated neural network's behavior is consistent with the behavior of experimental subjects in Costa Gomes et al. through the use of what seems like payoff dominance.[18] However, much like real people, as our neural network diverges from rational (Nash) behavior, it becomes increasingly difficult to tie down its underlying motivation; nevertheless, progress can be made. We suggest that as we more understand the methods used by biologically plausible neural networks, so we may better hope to understand the errors made by game-players in the laboratory and in the real world. An interesting next step for future research might be to combine neural networks with an evolutionary dynamic process or to explore best response learning to neural network behavior.[19]

References

Anthony, M., Bartlett, P.L., 1999. Neural Network Learning: Theoretical Foundations. Cambridge: Cambridge University Press.

Auer, P., Herbster, M., Warmuth, M.K., 1996. Exponentially many local minima for single neurons. In: Turetzky, D.S., Mozer, M.C., Hasslemo, M.E. (Eds.). Advances in Neural Information Processing Systems, vol. 8. Cambridge, MA. and London: MIT Press, 316-322.

Barr, J., Saraceno, F., 2005. Cournot competition, organization and learning. Journal of Economic Dynamics and Control 29, 277-295.

[18] Caution is required, of course, in interpreting this apparent consistency. It is of course possible that the success rates of the neural networks in facing the testing set in this paper and of the human subjects in Costa Gomes et al. are similar because they use similar local minimizing algorithms, or alternatively the similarity could be entirely coincidental. Please also note that we are not suggesting that the Costa Gomes et al.'s games can be used to train the network effectively (see footnote 11 for more details on this).

[19] For example, consider the following process. First, train a reinitialised neural network C^1 to select a best response to C^*. Then, train a reinitialised neural network C^2 to select a best response to C^1. Continue this iterative process with C^n trained to select a best response to C^{n-1} for large n. If the process converges we would have a Nash equilibrium in neural sets. Should the process not converge, the learning dynamic might still be informative.

Ben-Porath, E., 1990. The complexity of computing a best response automaton in repeated games with mixed strategies. Games and Economic Behavior 2, 1-12.

Camerer, C., Ho, T-H., 1999. Experience-weighted attraction learning in normal form games. Econometrica 67, 827-874.

Cho, I.K., 1994. Bounded rationality, neural networks and repeated games with discounting. Economic Theory 4, 935-957.

Cho, I.K., 1995. Perceptrons play the repeated Prisoner's Dilemma. Journal of Economic Theory 67, 266-284.

Cho, I.K., 1996. Perceptrons play repeated games with imperfect monitoring. Games and Economic Behavior 16, 22-53.

Cho, I.K., Sargent, T.J., 1996. Neural networks for encoding and adapting in dynamic economies. In: Amman, H.M., Kendrick, D.A., Rust, J. (Eds.). Handbook of Computational Economics, vol. 1. Amsterdam: North Holland, 441-470.

Cooper, D.J., Kagel, J.H., 2003. Lessons learned: Generalizing learning across games. American Economic Review 93, 202-207.

Costa Gomes, M., Crawford, V.P., Broseta, B., 2001. Cognition and behavior in normal-form games: An experimental study. Econometrica 69, 1193-1235.

Fantino, E., Stolarz-Fantino, S., 2005. Context and its effects on transfer. In: Zizzo, D.J. (Ed.). Transfer of Knowledge in Economic Decision Making. London: Palgrave Macmillan, 28-43.

Fukumizu, K., Amari, S., 2000. Local minima and plateaus in hierarchical structures of multilayer perceptrons. Neural Networks 13, 317-327.

Gilboa, I., 1988. The complexity of computing best response automata in repeated games. Journal of Economic Theory 45, 342-352.

Gilboa, I., Schmeidler, D., 2001. A Theory of Case-Based Decisions. Cambridge: Cambridge University Press.

Gilboa, I., Zemel, E., 1989. Nash and correlated equilibria: Some complexity considerations. Games and Economic Behavior 1, 80-93.

Ho, T-H., 1996. Finite automata play repeated prisoner's dilemma with information processing costs. Journal of Economic Dynamics and Control 20, 173-207.

Hornik, K., Stinchombe, M.B., White, H., 1989. Multi-layer feedforward networks are universal approximators. Neural Networks 2, 359-366.

Hutchins, E., Hazelhurst, B., 1991. Learning in the cultural process. In: Langton, C.G., Taylor, C., Farmer, J.D., Rasmussen, S. (Eds.). Artificial Life, vol. 2. Redwood City, California: Addison-Wesley, 689-705.

Leshno, M., Moller, D., Ein-Dor, P., 2003. Neural nets in a group decision process. International Journal of Game Theory 31, 447-467.

MacLeod, P., Plunkett, K., Rolls, E.T., 1998. Introduction to connectionist modelling of

cognitive processes. Oxford: Oxford University Press.

Macy, M., 1996. Natural selection and social learning in Prisoner's Dilemma: Co-adaptation with genetic algorithms and artificial neural networks. Sociological Methods and Research 25, 103-137.

Miller, J.H., 1996. The evolution of automata in the repeated prisoner's dilemma. Journal of Economic Behavior and Organization 29, 87-112.

Reilly, R., 1995. A connectionist exploration of the computational implications of embodiment. In: Proceedings of the Swedish Conference on Connectionism. Hillsdale: Lawrence Erlbaum, 237-258.

Roth, A.E., Erev, I., 1995. Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. Games and Economic Behavior 8, 164-212.

Roth, A.E., Erev, I., 1998. Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. American Economic Review 88, 848-881.

Rubinstein, A., 1993. On price recognition and computational complexity in a monopolistic model. Journal of Political Economy 101, 473-484.

Rumelhart, D.E., Hinton, R.J., Williams, R.J., 1986. Learning representations by backpropagating error. Nature 323, 533-536.

Rumelhart, D., McClelland, J., 1986. On learning the past tense of English verbs. In: McClelland, J., Rumelhart, D. (Eds.). Parallel Distributed Processing, vol. 2. Cambridge, MA: MIT Press, 216-271.

Sgroi, D., 2005. Using neural networks to model bounded rationality in interactive decision-making. In: Zizzo, D.J. (Ed.). Transfer of Knowledge in Economic Decision Making. London: Palgrave Macmillan, 128-147.

Sgroi, D., Zizzo, D.J., 2002. Strategy learning in 3x3 games by neural networks. Cambridge University Working Papers in Economics 0207.

Simon, H., 1955. A behavioral model of rational choice. Quarterly Journal of Economics 69, 99-118.

Simon, H., 1959. Theories of decision-making in economics and behavioral science. American Economic Review 49, 253-283.

Slonim, R., 1999. Learning rules of thumb or learning more rational rules. Journal of Economic Behavior and Organization 38, 217-236.

Sontag, E.D., 1995. Critical points for least-squares problems involving certain analytic functions with applications to sigmoidal nets. Advances in Computational Mathematics 5, 245-268.

Sontag, E.D., Sussmann, H.J., 1989. Backpropagation can give rise to spurious local minima even for networks with no hidden layers. Complex Systems 3, 91-106.

Stahl, D.O., 2000. Rule learning in symmetric normal-form games: Theory and evidence.

Games and Economic Behavior 32, 105-138.

Stahl, D.O., 2001. Population rule learning in symmetric normal-form games: Theory and evidence. Journal of Economic Behavior and Organization 45, 19-35.

Stahl, D.O., 2005. Action-reinforcement learning versus rule learning. In: Zizzo, D.J. (Ed.). Transfer of Knowledge in Economic Decision Making. London: Palgrave Macmillan, 44-72.

Stahl, D.O., Wilson, P.W., 1994. Experimental evidence on players' models of other players. Journal of Economic Behavior and Organization 25, 309-327.

Stahl, D.O., Wilson P.W., 1995. On players' models of other players: Theory and experimental evidence. Games and Economic Behavior 10, 218-254.

Weber, R.A., 2003. 'Learning' with no feedback in a competitive guessing game. Games and Economic Behavior 44, 134-144.

White, H., 1992. Artificial neural networks: approximation and learning theory. Cambridge and Oxford: Blackwell.

Zizzo, D.J., 2002. Neurobiological measurements of cardinal utility: Hedonimeters or learning algorithms? Social Choice and Welfare 19, 477-488.

Zizzo, D.J., 2005. Simple and compound lotteries: Experimental evidence and neural network modelling. In: Zizzo, D.J. (Ed.). Transfer of Knowledge in Economic Decision Making. London: Palgrave Macmillan, 166-193.

Zizzo, D.J., Sgroi, D., 2000. Bounded-rational behavior by neural networks in normal form games. Nuffield College Oxford Economics Discussion Paper 2000-W30.

Tables from "Learning to Play 3×3 Games: Neural Networks as Bounded-Rational Players"

	Correct Answer Given		
Convergence	**Error tolerance criterion**		
Level γ	**0.05**	**0.25**	**0.5**
0.1	**60.03**	**73.47**	**80**
0.05	**64.12**	**74.75**	**80.09**
0.02	**66.66**	**75.47**	**79.96**
Convergence	**Error tolerance criterion**		
Rate η	**0.05**	**0.25**	**0.5**
0.1	**63**	**74.48**	**80.22**
0.3	**63.3**	**74.42**	**79.89**
0.5	**63.66**	**74.54**	**79.94**
Convergence	**Error tolerance criterion**		
Rate μ	**0.05**	**0.25**	**0.5**
0	**62.86**	**74.63**	**80.47**
0.3	**62.89**	**74.49**	**80.22**
0.6	**63.73**	**74.53**	**79.9**
0.9	**64.05**	**74.05**	**79.04**

Table 1: Percentage of Correct Answers

	Correct Answer Given Error tolerance criterion		
	0.05	**0.25**	**0.5**
Trained *C**	**63.31**	**74.48**	**80.02**
Null1 (Untrained *C*)	**0**	**0**	**0**
Null2 (Strategy Switcher)	**22.8**	**22.8**	**22.8**
Null3 (Random)	**0.003**	**0.06**	**25.5**

Table 2: Average Performance of the Trained Network versus Three Null Hypotheses

Non-Nash Algorithm	Answerable Games (%)	Uniquely predicted Nash equilibria (%)	Expected performance in predicting Nash (%)
0 Level Strict Dominance	**18.3**	**18.3**	**46.97**
1 Level Strict Dominance	**39.75**	**39.75**	**62.19**
Rationalizability	**59.35**	**59.35**	**74.78**
L1	**100**	**67**	**67**
L2	**99.75**	**88.4**	**88.48**
Maximum Payoff Dominance	**99.6**	**62.1**	**62.23**
Minmax	**99.75**	**58.55**	**58.63**
Nearest Neighbor	**100**	**62.8**	**62.8**

Table 3: Answerable Games and Relationship to Nash

Table 3 footnote: Answerable games are games for which the Nash algorithm provides one, and exactly one, solution. The percentage is equal to (Number of Answerable Games)/2000. The central column considers the cases where these unique solutions coincide with the pure Nash equilibrium of the game. The right column adjusts this Nash predictive success of the non-Nash algorithm by making an auxiliary assumption on the agent's play in games where the non-Nash algorithm does not provide a unique solution: namely, we assume that the agent randomizes over all the choices (two or three, according to the game) admissible according to the non-Nash algorithm (e.g., in the case of rationalizability, all the non-dominated solutions).

Algorithm	% of Correct Answers Over Answerable Games		% of Correct Answers Over Full Testing Set		Overall Performance Over Full Testing Set (%)	
	=0.25	=0.05	=0.25	=0.05	=0.25	=0.05
Nash	**75.7**	**66.53**	**75.7**	**66.53**	**75.7**	**66.53**
L2	**82.23**	**76.54**	**82.03**	**76.35**	**82.11**	**76.44**
L1	**63.48**	**55.97**	**63.48**	**55.97**	**63.48**	**55.97**
Rationalizability	**82.57**	**74.36**	**49.01**	**44.13**	**61.94**	**57.18**
Nearest Neighbor	**58.78**	**51.48**	**58.78**	**51.48**	**58.78**	**51.48**
MPD	**57.02**	**49.95**	**56.79**	**49.75**	**56.79**	**49.76**
Minmax	**53.83**	**46.89**	**53.7**	**46.77**	**53.77**	**46.85**
1SD	**84**	**76.25**	**33.39**	**30.31**	**53.15**	**50.19**
0SD	**87.79**	**80.98**	**16.06**	**14.82**	**43.35**	**42.12**

Figure 1: Table 4: Describability of C^*'s Behavior by Non-Nash Algorithms

Table 4 footnote: The % of correct answers over answerable games = (number of correct answers) / (number of answerable games). Correct answers here implies giving the same answer as C^*. Answerable games are games for which the algorithm identifies a unique solution. % of correct answers over full testing set = (number of correct answers) / (number of answerable games). Expected performance over full testing set: % of correct answers over full testing set + adjustment due to the assumption of randomization over admissible actions in non-answerable games.

Action	Game A (ε = 0)			Game B (ε = 0.050)		
1	0.189, 0.142	0.379, 0.888	0.546, 0.427	0.554, 0.792	0.647, 0.015	0.759, 0.639
2	0.559, 0.268	0.033, 0.909	0.39, 0.8	0.697, 0.004	0.96, 0.22	0.911, 0.133
3	0.208, 0.337	0.991, 0.843	0.986, 0.85	0.998, 0.203	0.305, 0.282	0.686, 0.491
Action	**Game C (ε = 0.460)**			**Game D (ε = 0.991)**		
1	0.736, 0.825	0.335, 0.234	0.149, 0.047	0.965, 0.978	0.537, 0.262	0.301, 0.99
2	0.977, 0.747	0.993, 0.301	0.308, 0.844	0.166, 0.037	0.014, 0.018	0.754, 0.171
3	0.437, 0.805	0.873, 0.471	0.727, 0.939	0.054, 0.886	0.13, 0.157	0.283, 0.072

Table 5: Examples of C^*'s Performance, in Terms of Root Mean Square Error Levels ε, in Four Games from the Training Set